REMARKS

Claims 1-6 and 8-14 stand rejected under 35 USC §103(a) as being unpatentable over Bickle et al. publication entitled "Differential Effective Lapse Time Accumulator (Delta) in view of "How Debuggers Work" by Rosenberg in view of Dreyer et al., U.S. Patent 5,657,253, in view of Carter et al., U.S. Patent 6,249,907. Claim 7 stands rejected under 35 USC §103(a) as being unpatentable over Bickle and Rosenberg, Dreyer and Carter further in view of Hawley et al, U.S. patent 5,533,192.

Applicants respectfully submit that the independent claims 1, 6, and 11 clearly distinguish over the prior art references of record, and each of the independent claims 1, 6, and 11 is patentable.

Reconsideration and allowance of each of the pending claim 1-2, 4-11, and 13-14, as presented, is respectfully requested.

The Scope and Content of the prior art

The Bickle et al. publication entitled Differential Effective Lapse Time Accumulator (DELTA) discloses a test tool, DELTA, applied to processors whose address bus and control signals are defined as conditions and timing measurements are taken between address (breakpoint) A and address B or to count the number of times the processor executes a particular bus state. The tool allows the measurement of system performance in two areas: (1) instruction cycle time measurement and (2) program execution time, instruction count and busy state time measurement, under the control of a microprocessor 12. The Bickle et al. publication discloses a display terminal 25 providing an operator interface that is used to enter breakpoint and oscillator

parameters and to display timing/counting results under the control of the microprocessor 12.

"How Debuggers Work" by Rosenberg discloses hardware debugger facilities, at pages 39-40 states the mimimum basic requirements a debugger place on underlying hardware are 1. A way to specify a breakpoint such that when the processor reaches this location, execution will stop. 2. A notification system, also called an interrupt or a trap, that will notify the operating system (and thereby the debugger) that an important event has occurred with respect to the running process. 3. The ability to read and write directly out of and into the hardware registers when the interrupt occurs; this includes the program counter register.

Dreyer et al., U.S. Patent 5,657,253 discloses an apparatus for measuring and monitoring various parameters that contribute to the performance of a processor that includes a pair of programmable event counters for counting any two independent events selected from a predetermined list of processor events. A specialized register controls the operation of the event counters and also selects the events to be counted. The contents of the event counters can be accessed either by a supervisor mode program which reads an instruction or through a special access port.

The newly cited Carter et al., U.S. Patent 6,249,907 discloses a system for debugging a computer program. A user indicates a specified breakpoint type, such as a program statement, variable reference, command, and the like. The program, including program statements, is then compiled. During compilation, the compiler locates statements in the program corresponding to the breakpoint types and generates a

function call into the program at instances in the program of statements corresponding to the user specified breakpoint types. During a debugging phase, a debugger may execute an executable version of the program, including the function calls. Upon processing the function calls, the debugger may stop execution of the program and pass control to the user to perform debugging operations. As stated at column 4, lines 35-52 and column 5, lines 4-15:

> The debugger 16 and program when executing may communicate via the operating system 12. In the embodiments where the debugger 16 and the program execute in separate processes, the debugger 16 can control the execution of the object code 20 or obtain information on the execution of the object code 20 via system calls, i.e., APIs, to the operating system 12, which in turn directly controls the execution of the object code 20. In the embodiments where the debugger 16 and object code 20 execute within the same process, calls to the debugger 16 can be placed in the object code 20 so that during the execution of the object code 20, the object code 20 can call, via the operating system 12, subroutines of the debugger 16 to pass control to the debugger 16. Preferred embodiments of the present invention may be implemented in either environment, i.e., the execution of the debugger 16 and object code 20 may be in the same or different processes.
>
> * * *
>
> During compilation, the compiler 14 will generate breakpoint indicators or "hooks" into the object code 18 at each instance in the program of the statement corresponding to the user specified breakpoint type. Hooks are instructions the compiler 14 inserts into the program, usually the object code 20, during compilation. The hooks can be used to set breakpoints that instruct the debugger 16 to gain control of the program at specific points in the program. The hooks may be inserted at the entrances and exits of blocks, at statement boundaries, and at points in the program where program flow might change, such as before and after a procedure call.

Hawley et al, U.S. patent 5,533,192 discloses a program debugging system having a core unit that includes a plurality of debugger memory areas, each uniquely associated with a corresponding one of a plurality of debuggers. The core unit responds to an exception condition by selecting one debugger from the plurality of debuggers, selection being made by determining which one of the debuggers is

associated with the program exception. Then, computer state information and debugger state information are stored into a selected one of the debugger memory areas that is exclusively associated with the selected debugger, and the selected debugger is activated. A new debugger may register with the core unit, so that the new debugger is added to the plurality of debuggers. The activated debugger may send a debugging command to the core unit, which responds by updating debugger state information based on the received debugging command, and storing the updated debugger state information into the selected debugger memory area. When a debugger relinquishes control of the computer, the core unit retrieves the updated debugger state information from the selected debugger memory area, and controls the hardware resources in accordance therewith. If the updated debugger state information includes an indication that a breakpoint is set, the core unit sets a breakpoint that includes information associating the set breakpoint with the selected debugger. When the breakpoint is triggered, the core unit identifies from the breakpoint information which of the debuggers the breakpoint is associated with, and activates the identified debugger.

The present invention

The present invention enables improved timing of a particular function, and to count selected programmable events during the execution of a function with a breakpoint manager and performance measurement program in accordance with the preferred embodiment. The present invention enables programmable processor event counters to start counting immediately at the beginning of this bounded call flow and stop counting immediately at the end of the bounded call flow. The bounds of the call

flow are provided at arbitrary locations within the instruction stream.

The present invention provides code or breakpoint handler instructions allows the user of the debugger 134 to interrogate the program state and most importantly allows the user to request a return to program processing. The breakpoint handler does this by ultimately executing a single instruction indicated at a line labeled START PROCESSING that returns processing out of the interrupt handler and into the program again, immediately after the point of the first breakpoint interrupt, as indicated at a line labeled START RETURN-FROM-INTERRUPT INSTRUCTION from interrupt handler instructions 206 to the compiler-generated instructions 204. The reason this is important to the breakpoint performance measurement method of the preferred embodiment is that at this point in time, the initialized programmable hardware counters 140 are enabled to begin counting. It is this point in time that defines the beginning of the bounded instruction stream. Now the user's program is executing, calling arbitrary routines, using arbitrary processor facilities, and counting arbitrary processor events. At some point in time execution reaches the ending bound of the measurement period.

The value in a breakpoint instruction in accordance with the preferred embodiment essentially selects which of the possible types of events are to be counted with the programmable processor event counters.

Applicants respectfully submit that each of the pending claims 1-2, 4-11, and 13-14, as presented, is patentable, when correctly considering (1) the scope and content of the prior art, (2) the differences between the claimed invention and the prior art, (3) the level of ordinary skill in the pertinent art, and (4) objective evidence relevant

to the issue of obviousness.

Applicants respectfully submit that one of ordinary skill in the art would not have been led to the claimed invention by the reasonable teachings or suggestions found in the prior art, including the Bickle, Rosenberg, Dreyer et al., and Hawley references, as now recited in independent claims 1, 6, and 11, as presented.

Independent claim 1, as presented, recites the step of providing a debugger breakpoint manager including a performance measurement program and a user interface, and enabling a user to specify a start bound and an end bound of a performance collection region of a user source code and said set of hardware counters.

Each of the independent claims 1, and 11, as presented, recite the steps of providing compiler-generated hardware instructions defining breakpoint instructions within an instruction stream; said compiler-generated hardware instructions including a start breakpoint instruction and a stop breakpoint instruction; inserting said start breakpoint instruction and said stop breakpoint instruction in compiler-generated hardware instructions for a user source code. Independent claim 6, as presented, recites a compiler generating hardware instructions defining breakpoint instructions within an instruction stream; said compiler-generated hardware instructions including a start breakpoint instruction and a stop breakpoint instruction and inserting said start breakpoint instruction and said stop breakpoint instruction in compiler-generated hardware instructions for a user source code.

Each of the independent claims 1, and 11, as presented, further recite executing said compiler-generated hardware instructions and suspending processing of

said compiler-generated hardware instructions responsive to executing said start breakpoint instruction; responsive to executing said start breakpoint instruction generating a processor interrupt for entering interrupt handler instructions and calling breakpoint instructions; said breakpoint instructions generating a start processing instruction to return processing from said interrupt handler instructions to said compiler-generated hardware instructions and starting said defined set of hardware counters, responsive to said generated start processing instruction; executing said compiler-generated hardware instructions and suspending processing of said compiler-generated hardware instructions and stopping said defined set of hardware counters, responsive to executing said stop breakpoint instruction. Independent claim 6, as presented, recites user program means for executing said compiler-generated hardware instructions and suspending processing of the hardware instructions responsive to executing said start breakpoint instruction and generating a processor interrupt for entering interrupt handler instructions and for calling said breakpoint manager; said breakpoint manager for generating a start processing instruction to return processing from said interrupt handler instructions to said compiler-generated hardware instructions and starting said defined set of hardware counters, responsive to said generated start processing instruction; and said user program means for executing said compiler-generated hardware instructions and suspending processing of the hardware instructions and stopping said defined set of hardware counters, responsive to executing said stop breakpoint instruction.

As recited in independent claims 1, 6, and 11, as presented, the subject

matter of the invention is patentable over the references of record including the Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references. The present invention provides enhanced breakpoint based performance measurement that is different from the prior art including the Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley.

Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references fail to suggest inserting a start breakpoint instruction and a stop breakpoint instruction in hardware instructions; executing said hardware instructions and suspending processing of said hardware instructions responsive to executing said start breakpoint instruction; and responsive to executing said start breakpoint instruction generating a processor interrupt for entering interrupt handler instructions and calling breakpoint instructions, and that said breakpoint instructions generating a start processing instruction to return processing from said interrupt handler instructions to the hardware instructions and starting said defined set of hardware counters, responsive to said generated start processing instruction, as taught and claimed by applicants.

The Examiner acknowledges that Bickle does not disclose suspending processing of said hardware instructions responsive to executing said start breakpoint instruction. Rosenberg teaches that mimimum basic requirements a debugger place on underlying hardware are a way to specify a breakpoint such that when the processor reaches this location, execution will stop, a notification system, also called an interrupt or a trap, that will notify the operating system (and thereby the debugger) that an important event has occurred with respect to the running process, and the ability to read and write directly out of and into the hardware registers when the interrupt occurs; this

includes the program counter register. Rosenberg and Dreyer also fail to suggest executing said hardware instructions and suspending processing of said hardware instructions responsive to executing said start breakpoint instruction.

The Examiner now cites Carter reference and maintains that that Carter teaches that compilers generate hardware instructions. Applicants respectfully submit that the teachings of Carter are different from and fail to achieve executing said hardware instructions and suspending processing of said hardware instructions responsive to executing said start breakpoint instruction. The compiler of Carter generates object code; however Applicants respectfully submit that Carter does not teach generating hardware instructions as taught and claimed by Applicants, when the claim term is construed in a manner consistent with the ordinary and customary meaning of the claim term and as taught by Applicants the patent specification where the invention is described.

Claims should be given their broadest reasonable interpretation consistent with the specification. The claim language itself governs its meaning. Vitronics Corp. v. Conceptronic, Inc., 90 F.3d 1576, 1582 (Fed. Cir. 1996). The meaning of claim language is construed according to its usage and context. ResQNet.com, Inc. v. Lansa, Inc., 346 F.3d 1374, 1378 (Fed. Cir. 2003). The touchstone for discerning the usage of claim language is the understanding of those terms among artisans of ordinary skill in the relevant art at the time of invention. See Rexnord Corp. v. Laitram Corp., 274 F.3d 1336, 1342 (Fed. Cir. 2001). Indeed, normal rules of usage create a "heavy presumption" that claim terms carry their accustomed meaning in the relevant

community at the relevant time. CCS Fitness, Inc. v. Brunswick Corp., 288 F.3d 1359, 1366 (Fed. Cir. 2002) (citing Johnson Worldwide Assocs., Inc. v. Zebco Corp., 175 F.3d 985, 989 (Fed. Cir. 1999)). The best source for discerning the proper context of claim terms is the patent specification wherein the patent applicant describes the invention. In addition to providing contemporaneous technological context for defining claim terms, the patent applicant may also define a claim term in the specification "in a manner inconsistent with its ordinary meaning." Boehringer Ingelheim Vetmedica, Inc. v. Schering-Plough Corp., 320 F.3d 1339, 1347 (Fed. Cir. 2003) (citing Teleflex, 299 F.3d at 1325-26). Dictionaries, encyclopedias and treatises are particularly useful resources to assist in determining the ordinary and customary meanings of claim terms. Tex. Digital Sys., Inc. v. Telegenix, Inc., 308 F.3d 1193, 1202 (Fed. Cir. 2002).

The references of record, including Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley fail to suggest these recited claim limitations, and further that said breakpoint instructions generating a start processing instruction to return processing from said interrupt handler instructions to the hardware instructions and starting said defined set of hardware counters, responsive to said generated start processing instruction, and executing said compiler-generated hardware instructions and suspending processing of said compiler-generated hardware instructions and stopping said defined set of hardware counters, responsive to executing said stop breakpoint instruction, as taught and claimed by applicants.

Applicants respectfully submit that each of the independent claims 1, 6, and 11, as presented, is patentable in view of the differences between the claimed

invention and the prior art.

Applicants respectfully submit that the total teachings of Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references, would not enable a user to specify a start bound and an end bound of a performance collection region of a user source code, nor achieve the programmable hardware counters, nor that the set of hardware counters is specified by the user, as taught and claimed by Applicants.

Applicants respectfully submit that one of ordinary skill in the art would not have been led to the claimed invention by the reasonable teachings or suggestions found in the prior art, including the Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references, as now recited in independent claims 1, 6, and 11, as presented.

Applicant respectfully submits that the combined teachings of the Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references themselves, when considered with the knowledge generally available to one of ordinary skill in the art, fail to teach or suggest all the claim limitations as now recited in independent claims 1, 6, and 11, as presented.

None of the Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references disclose or remotely suggest the programmable processor hardware counters, and providing a debugger breakpoint manager including a performance measurement program and a user interface, and enabling a user to specify a start bound and an end bound of a performance collection region of a user source code and said set of hardware counters, as defined by independent claims 1 and 6 to include processor cycles and cache misses, or by independent claim 11 to include processor

cycles and translation lookaside buffer misses, nor the steps of starting said defined set of hardware counters, responsive to said generated start processing instruction; and executing the hardware instructions and suspending processing of the hardware instructions and stopping said defined set of hardware counters, responsive to executing said end breakpoint instruction, and enabling a user to specify a start bound and an end bound of a performance collection region of a user source code and said set of hardware counters, as taught and recited in each of the independent claims 1, 6, and 11, as presented.

Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references fail to suggest the programmable processor hardware counters for counting predefined programmable processor events including processor cycles and cache misses, as recited in independent claims 1, and 6. Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references fail to suggest the programmable processor hardware counters for counting predefined programmable processor events including processor cycles and translation lookaside buffer misses, as recited in independent claim 11.

Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references fail to suggest providing a debugger breakpoint manager including a performance measurement program and a user interface, as recited in independent claims 1, and 6. Bickle, Rosenberg, Dreyer et al., and Hawley references fail to suggest enabling a user to specify a start bound and an end bound of a performance collection region of a user source code and said set of hardware counters, as recited in independent claims 1, 6, and 11, as presented.

Applicants respectfully submit that Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley, considering the total teachings in combination, fail to suggest the subject matter and the above recited features of the present invention as set forth in independent claims 1, 6, and 11, as presented.

Thus, each of the independent claims 1, 6, and 11, as presented, is patentable.

Dependent claims 2, 4-5, 7-10, and 13-14, as presented, respectively depend from patentable claims 1, 6, and 11, further defining the invention. Each of the dependent claims 2, 4-5, 7-10, and 13-14, as presented, is likewise patentable.

Applicants have reviewed all the art of record, and respectfully submit that the claimed invention is patentable over all the art of record, including the references not relied upon by the Examiner for the rejection of the pending claims.

It is believed that the present application is now in condition for allowance and allowance of each of the pending claims 1-2, 4-11, and 13-14, as presented, is respectfully requested. Prompt and favorable reconsideration is respectfully requested.

If the Examiner upon considering this amendment should find that a telephone interview would be helpful in expediting allowance of the present application, the Examiner is respectfully urged to call the applicants' attorney at the number listed below.

Respectfully submitted,

S-signature by

___/Joan Pennington/________

By: Joan Pennington
Reg. No. 30,885
Telephone: (312) 670-0736

March 5, 2008